APPLICATION FOR PATENT

TITLE: OPERATING SYSTEM-INDEPENDENT COMPUTING SYSTEM USER FEEDBACK MECHANISM

INVENTORS: JOHN A. LANDRY, VALIUDDIN Y. ALI, AJAY CHATURVEDI, BROOKS A. RORKE, STACY L. WOLFF, and KEVIN L. MASSARO

SPECIFICATION

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation-in-part of U.S. patent application Serial No. 09/478,153, entitled "DIGITAL FEEDBACK DISPLAY PANEL AND SUPPORTING SOFTWARE FOR A COMPUTER USER," filed January 5, 2000, which is incorporated herein by reference in its entirety for all purposes.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention generally relates to operating condition user feedback for computing systems, and more particularly to an operating system-independent user feedback mechanism of a computing system.

2. Description of the Related Art

User feedback with respect to operating conditions or states of a computer system has typically been provided by a monitor, audio alerts, light emitting diodes (LEDs) or directly by an operating system of the computer system. Each of these forms of user feedback in effect depends upon operating system control. Where an operating condition relates to an error with the operating system itself, the operating system cannot detect such an error. A lock-up of the operating system is therefore particularly difficult for a user to detect. One drawback of relying upon the operating system and the monitor to provide user feedback is that neither the operating system nor the monitor are available at all times. A computer system is typically configured to enter a low power or sleep mode after a certain period of idle time where neither the monitor nor the operating system are available. Because of the unavailability of the

monitor and the operating system in a low power mode, certain operating conditions are not communicated to a user. For example, a user cannot determine when a new e-mail is available during a low power mode. Even a feature such as a digital clock has depended upon the monitor being fully awake and upon the operating system.

Many computer users tend to be frustrated with user feedback in the form of audio alerts and LEDs. Audio alerts are perceived by users as annoying. Many users therefore lower the volume of their computer systems, defeating the purpose of the audio alerts. LEDs (including the hard disk drive LED of a computer system) are perceived by users as confusing, particularly where flashing patterns of the LEDs signify different operating conditions of the computer system. Even when users somewhat understand an operating condition conveyed by the monitor, audio alerts or LEDs, most users are fearful of touching and inadvertently "crashing" or depowering their computer systems, thereby risking a loss of data.

The operating system, monitor, audio alerts and LEDs alike have provided little, if any, feedback to users during critical times for a computer system such as system initialization or power-up, for example. Users tend to ignore or overlook any system information which is quickly flashed on the monitor during power-up. Users also have difficulty detecting when a computer system is in fact connected to the Internet. In the case of a user who frequently moves his or her desktop computer, such a user may be forced to check one or more cable connections if the computer does not power-up after the computer is relocated. Users thus are typically not well informed of the operating conditions or events of computer systems, if at all, and must resort to contacting the computer manufacturer, contacting the Internet service provider, examining the user's manual or examining the computer itself when a perceived problem develops.

SUMMARY OF THE INVENTION

Briefly, a computing system employs a user feedback mechanism to monitor a plurality of operating conditions of the computing system and to alert a user to the plurality of operating conditions independently of an operating system of the computing system. The user feedback mechanism includes a display panel to display a plurality of operating condition messages to the user and includes a controller to monitor a plurality of operating condition signals. Further, the user feedback mechanism is independently powered and further includes a safety button to signal a power supply to power off the computing system independently of the operating system. Examples of operating conditions include a connection state of the

computing system to the Internet and a connection state of a peripheral device to the computing system. Other examples of operating conditions that the user feedback mechanism displays to the user independently of the operating system include a new e-mail notification message, a new Internet message or atomic time from a network server coupled to the computing system. An operating condition message is cleared from the display panel when the operating condition is cured.

BRIEF DESCRIPTION OF THE DRAWINGS

A better understanding of the present invention can be obtained when the following detailed description of the preferred embodiment is considered in conjunction with the following drawings, in which:

Figure 1 is a block diagram of an exemplary software architecture of a computing system with an operating system-independent user feedback module;

Figure 2 is a block diagram of an exemplary hardware architecture of the computing system with an operating system-independent user feedback mechanism;

Figures 3A and 3B show a flow chart illustrating exemplary operation of the operating system-independent user feedback module of Figure 1 and the operating system-independent user feedback mechanism of Figure 2;

Figure 4 is a flow chart illustrating exemplary processing of the safety button of the operating system-independent user feedback mechanism of Figure 2;

Figure 5 is a block diagram of an exemplary architecture of the operating system-independent user feedback module of Figure 1; and

Figure 6 is a block diagram illustrating exemplary application-level interfacing and system-level interfacing for the operating system-independent user feedback mechanism of Figure 2.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENT

The following commonly-assigned patent application is incorporated herein by reference in its entirety for all purposes:

U.S. Patent Application, Serial No. 09/557,700, entitled "OPERATING SYSTEM HANG DETECTION AND CORRECTION," filed April 25, 2000 by Craig L. Chaiken and Stan Stanart.

Turning now to the drawings, Figure 1 shows exemplary computing system software CS including an operating system 100, system BIOS (Basic Input Output System) 102, an

operating system-independent user feedback or interface module 106 and an LCD (liquid crystal display) interface driver 118. The operating system 100 is a typical operating system including a system registry 126, a clock 120, an Internet manager 124 and a task scheduler 122. The operating system 100 is coupled to both system BIOS 102 and the user feedback module 106. The user feedback module 106 includes an operating system interface 110 coupled to the operating system 100, a BIOS interface 112 coupled to the system BIOS 102 and an ACPI (Advanced Configuration and Power Interface) interface 114 to interface with ACPI logic. The LCD interface driver 118 is also coupled to the user feedback module 106. The user feedback module 106, the LCD interface driver 118 and the system BIOS 102 in combination provide software control of an operating system-independent user feedback or digital dashboard mechanism 200 such as shown in Figure 2. The term "computing system" as used herein refers to any system with a superset or subset of processing or Internet functions of a computer system, including but not limited to Internet appliances and information appliances.

Referring to Figure 2, exemplary computing system hardware CH including the user feedback mechanism 200 is shown. The user feedback mechanism 200 is employed to monitor operating conditions (including fault conditions) of a computing system and to alert a user to the operating conditions independently of the operating system 100. The concept of "monitoring" operating conditions should be construed to encompass detecting data and processing the data to a more meaningful form to determine an operating condition. An operating condition generally refers to any event or condition related to the operation of the computing system, including but not limited to fault conditions of the computing system. The term "fault condition" should be construed to encompass any system condition that may negatively impact a user's experience with the computing system. The operating system-independent user feedback mechanism 200 includes a mini display or LCD 204, an interface controller 202, a message indicator 208, and a safety button 206. The interface controller 202 is shown coupled to the mini LCD 204, the message indicator 208, and the safety button 206. A user is alerted of operating conditions of the computing system by the mini LCD 204 displaying operating condition messages. The mini LCD 204 receives an operating condition signal from the interface controller 202 and then displays a corresponding operating condition message. A variety of operating condition signals and operating condition messages may be supported by the user feedback mechanism 200. In addition to displaying operating condition messages, the mini LCD 204 may also display instructions to a user for the user to cure the particular operating condition. It should be understood that the mini display 204 may be

implemented with display technologies other than LCD. The message indicator 208, which for example may be implemented as one or more light-emitting diodes (LEDs), is used to indicate messages to a user such as an email message or an Internet message. The term "Internet message" refers to any non-email message communicated over the Internet, such as messages unique to a particular Internet service provider. The safety button 206 is used to signal a power supply 214 to power off the computing system if the computing system is not already powered off by the operating system 100. Processing of the safety button 206 is described in more detail below in connection with Figure 4.

The interface controller 202 is further coupled to ACPI logic 210. The ACPI logic 210, which is shown including an ACPI timer 212, is coupled to the power supply 214. The use of the ACPI timer 212 is described below in connection with Figure 4. The arrowed line from the power supply 214 to the user feedback mechanism 200 represents that the user feedback mechanism 200 is powered independently of the rest of the computing system. When the user feedback mechanism 200 is powered (i.e., when the user feedback mechanism power cable is plugged in), the mini LCD 204 is lit. Unlike the computing system itself, the user feedback mechanism 200 is preferably left on or powered at all times. The user feedback mechanism 200 thus operates even when the computing system is off or asleep. Other illustrated hardware of the computing system includes a main processor 216, a monitor 220, a keyboard 222, a mouse 223 and a peripheral device 224. The main processor 216 is a typical microprocessor for running an operating system and other software. The monitor 220, the keyboard 222, the mouse 223 and the peripheral device 224 represent peripheral devices which may generate operating conditions to be detected and then displayed to a user. For example, if the monitor 220 is disconnected, a message (e.g., text and/or icon) indicating the monitor 220 is disconnected may be displayed on the mini LCD 204. In a disclosed embodiment, the user feedback mechanism 200 monitors each level of connection and removal of peripheral devices of the computing system. The monitor 220, the keyboard 222 and the mouse 223 are treated as primary devices of the computing system. As such, the states of these devices are checked during system initialization as described in more detail below.

Referring to Figures 3A-3B, an exemplary operating system-independent operating condition feedback process using the user feedback mechanism 200 is shown. The process begins at any of the steps 302-318 which represent detection of a variety of operating conditions. In step 302, the operating condition is an Internet connection state of the computing system. In step 304, the operating condition is the keyboard connection state of

the computing system. In step 306, the operating condition is the monitor connection state of the computing system. In step 308, the operating condition is the connection state of a peripheral device in the computing system. In step 310, the operating condition is a lock-up of the operating system 100. In step 312, the operating condition is an email notification. In step 314, the operating condition is an Internet message. In step 316, the operating condition is the atomic time. The computing system may detect atomic time from a network server during an Internet connection. In a disclosed embodiment, a default mode of the user feedback mechanism 200 is a clock mode in which atomic time is displayed on the mini LCD 204. Compared to a typical operating system-based clock of a computer system, the clock mode of the user feedback mechanism 200 runs independently of the operating system 100 and provides a more accurate clock. Even if a user does not initiate connection to the Internet every day, the user feedback mechanism 200 automatically connects to the Internet each day to ensure the accuracy of its clock. This automatic Internet connection feature may be configured by a user to force an Internet connection as many times a day as desired and at whatever times are desired. Since an OS-based clock is at times not visible on a monitor, users have typically relied upon clocks external to their computer systems. A user, however, can view the clock of the user feedback mechanism 200 at any time. In step 318, the operating condition is the power state (i.e., sleep mode) of the computing system. It should be understood that the operating conditions shown in steps 302-318 are illustrative and not exhaustive. As one example, an operating condition which may be detected using a watermark setting is whether a hard disk drive of the computing system is almost full. As another example, it may be detected whether the hard disk is about to crash. An operating condition may be detected by a notification or alert. Alternatively, at a desired time, the process may poll or query for an operating condition. For each of steps 302-318, control proceeds next to step 320. In step 320, the operating condition is communicated to the mini LCD 204. Communication of an operating condition to the mini LCD 204 is described below in connection with Figure 6. As represented by the step 328, a detected operating condition may be read by an application or the operating system 100.

Next, control proceeds to step 324 where an operating condition message corresponding to the operating condition is displayed by the mini LCD 204. From step 324, control proceeds to step 325 where an operating condition display complete signal is generated by the interface controller 202. Control next proceeds to step 326 where an instruction to cure the operating condition is displayed by the mini LCD 204. Next, in step 330, it is then determined if the operating condition is cured. If the operating condition is not

cured, then control remains in step 330. In this way, the instruction to cure the operating condition remains displayed until the operating condition is cured by the user. If it is determined in step 330 that the operating condition is cured, then control proceeds to step 332 where the instruction is cleared from the mini LCD 204. It should be understood that step 326 of displaying an instruction to cure a operating condition is optional. For example, if the operating condition message itself implies what a user needs to do to cure an operating condition, it may be desirable to not display an instruction to inform the user how to cure the operating condition. In other words, there may be some cases where once a user is informed of the operating condition, it is self-explanatory how to cure the operating condition.

Steps 334-340 are presented to illustrate situations where steps in addition to message display are performed by the user feedback mechanism 200 after detecting certain operating conditions. If an Internet message or an email notification is detected in step 334, control proceeds to step 336 where the message indicator 208 is set. For example, if two email notifications were detected, then the message indicator 208 is set to periodically blink twice. If an Internet message or email notification is not detected in step 334, then control proceeds directly to step 338. In step 338, it is determined if atomic time is detected. If not, control proceeds to step 342 indicating the operating condition feedback process is complete. If atomic time is detected in step 338, then control proceeds to step 340 where the clock 120 of the operating system 100 and a clock of the user feedback mechanism 200 are both updated with the atomic time. By updating the clock of the user feedback mechanism 200, the mini LCD 204 displays atomic time. From step 340, the process is completed in step 342.

Referring to Figure 4, processing of the safety button 206 (Figure 2) is shown. The process begins in step 400 where it is determined if a press or actuation of the safety button 206 by a user is detected. If a press or actuation is not detected, then control remains in step 400. If a press or actuation is detected, then control proceeds to step 402 where the ACPI timer 212 is started. Next, in step 404 a shutdown of the operating system 100 is initiated. Control then proceeds to step 406 where it is determined if the ACPI timer 212 has expired. If the ACPI timer 212 has not expired, then control remains in step 406. If the ACPI timer 212 has expired, then control proceeds to step 408 where it is determined if the operating system 100 is shutdown. Detection of a hang up by the operating system 100 is described in a commonly-assigned U.S. patent application, entitled "OPERATING SYSTEM HANG DETECTION AND CORRECTION," previously incorporated herein. If the operating system 100 is shutdown, then control proceeds to step 412 where processing of the safety button 206 is complete. If the operating system 100 is not shutdown, then control proceeds to

step 412 where the ACPI logic 210 is reset signaling the power supply 214 to turn off. At this point, the computing system can be considered as in a "safe mode." From step 410, control proceeds to step 412 where the processing of the safety button 206 is complete. This process provides a way to ensure that the operating system 100 is shutdown. By using the safety button 206, a user can be ensured that the computing system is in a depowered state. With this assurance, a user can then be more comfortable opening or moving the computing system. The use of the safe mode can prevent a user from accidentally powering off the computing system with files open in the operating system 100.

Referring to Figure 5, exemplary software components which may be included in the user feedback module 106 of Figure 1 is shown. As previously shown in Figure 1, the user feedback module 106 includes an operating system interface 110, the BIOS interface 112 and the ACPI interface 114. The user feedback module 106 may further include client software components such as a device monitor 500, an Internet monitor 502, an atomic time monitor 504, an email monitor 506 and an Internet message monitor 508. The device monitor 500 is used to detect the connection or other state of peripheral devices. For example, if the peripheral device is a CD-ROM drive, it can be detected whether the drive contains an audio CD, a data CD, or a DVD CD. In a disclosed embodiment, the device monitor 500 uses the WM_DEVICECHANGE signal provided by the operating system 100 to detect device states. If the device is a printer, then the device monitor 500 uses the WM_SPOOLERSTATUS signal provided by the operating system 100 to detect if a print job is started or finished. Those skilled in the art are familiar with calling application programming interfaces (APIs) into an operating system to retrieve desired information known by the operating system. A variety of standard APIs may be employed to retrieve potentially useful information for a user from an operating system. In the past, such information has not been retrieved, processed and displayed to the user.

The Internet monitor 502 is used to detect if the computing system is connected to the Internet. In a disclosed embodiment, the Internet monitor 502 registers with the operating system 100 to receive messages regarding Remote Access Services (RAS). The Internet monitor 502 is informed by a notification or message when an Internet connection is made or disconnected. The atomic time monitor 504 is used to detect atomic time from a server or network computer coupled to the computing system during an Internet connection. The email monitor 506 is used to detect email notifications for the computing system. The Internet message monitor 508 is used to detect Internet messages for the computing system. The monitors 500-508 correspond to steps 302-308 and 312-316 of Figure 3A. The monitors 500-

506 are described in more detail in the commonly-assigned U.S. patent application, entitled "DIGITAL FEEDBACK DISPLAY PANEL AND SUPPORTING SOFTWARE FOR A COMPUTER USER," previously incorporated herein.

Referring to Figure 6, exemplary application-level interfacing and system-level interfacing for the user feedback mechanism 200 is shown. Application-level interfacing relates to an application 602 or 604 communicating with the mini LCD 204. System-level interfacing relates to a client 612 communicating with the mini LCD 204. An LCD interface 600 is shown coupled to the application 602, the application 604 and an LCD interface driver 118. In a disclosed embodiment, communication to the mini LCD 204 in the form of system-level interfacing overrides communication to the mini LCD 204 in the form of application-level interfacing. The application 602 or 604 may for example be a debug application to track the operating conditions detected by the user feedback mechanism 200. In a disclosed embodiment, the LCD interface 600 is a component object model (COM)-based interface called by the application 602 or 604 for communication with the mini LCD 204. The techniques for a COM interface to communicate with an application are well known in the art. COM is a well known paradigm for interaction among software components. The LCD interface 600 is used to provide the applications 602 and 604 an application-level logical interface to the mini LCD 204 of the user feedback mechanism 200. For example, the application 604 through the LCD interface 600 can provide operating conditions to the user feedback mechanism 200 and can also specially configure display by the mini LCD 204 for the particular application 604. The LCD interface 600 is specially configured to encapsulate functionality for the application 602 or 604 to communicate with the mini LCD 204 of the user feedback mechanism 200. The LCD interface 600 is basically an open mechanism for any application 602 or 604 known by or registered with the operating system 100 to communicate with the mini LCD 204. The LCD interface 600 may also be used by the application 602 or 604 to read information from the mini LCD 204. The application 602 or 604 can be written in any computer language (e.g., Java, C++ or HTML).

The table below shows a variety of interface methods for the LCD interface 600. Beside the name of each interface method shown, a general description of the interface method is provided.

Interface Method Name	Interface Method Description
SetClock	This method updates the time of the internal clock of the mini LCD and the system clock of the operating system.
SetClockEX	This method updates the date and time of the internal clock of the mini LCD and the system clock of the operating system.
SetLEDState	This method is used to set an LED for the feedback mechanism to an on, off or blinking state. The supported LEDs are a sleep LED, an Internet detect LED, a power LED, a message LED and a backlight LED.
DisplayText	This method is used to control parameters for displaying text on the mini LCD.
SimplyDisplayText	This method is used to display text on the mini LCD with default text display parameters.
SimplyScrollText	This method is used to scroll text on the mini LCD with default scrolling parameters.
GetLCDLongProperties	This method is used to fetch the properties of the mini LCD.
ClearText	This method clears the text display area of the mini LCD and displays the clock if no pending text is waiting to be displayed.
DisplayIcon	This method is used to control parameters for displaying icons on the mini LCD.
SimplyDisplayIcon	This method is used to display icons on the mini LCD with default icon display parameters.
BlinkIcon	This method is used to control blinking of icons on the mini LCD.
DisplayIconEX	This method is used to display icons on the mini LCD with blinking and timeout control.
ClearIcon	This method is used to clear an icon from the mini LCD.
LoadIcon	This method downloads an icon with its icon handle to the mini LCD.

Interface Method Name	Interface Method Description
LoadIconFromBitmap	This method downloads an icon from a bitmap to the mini LCD.
LoadIconFromByteArray	This method downloads an icon from a one-dimensional array to the mini LCD.
LoadIconFromPicture	This method extracts icon data from an OLE picture object for the mini LCD.
LoadFont	This method extracts font data from an OLE font object for the mini LCD.
LoadFontFrom ByteArray	This method downloads a font from a byte array to the mini LCD.
PlaySound	This method plays sound for the feedback mechanism with default sound parameters.
PlaySoundEX	This method controls parameters for playing sound for the feedback mechanism.

It should be understood that the COM interface methods disclosed herein are illustrative and not exhaustive.

A variety of application-specific messages may be displayed by the mini LCD 204. If the application for example is Microsoft Word®, then the mini LCD 204 may be used to indicate that Microsoft Word® is launching or to indicate that Microsoft Word® is terminating. The application 602 or 604 can be scheduled for launching by the operating system 100 and can be self-terminated when not needed. As another example, if the application 604 is a paging application, then the application 604 can signal the mini LCD 204 to display "paging mom" when the application 604 is being used to page the mother of the user. Five other examples of applications that may communicate with the user feedback mechanism 200 include a calculator application, a calendar application, an alarm application, a voice mail application, and a web-based diagnostic application for the computing system manufacturer to remotely reply to service requests from the user or remotely download code updates to the user. Unlike application 604, application 602 is shown coupled to the operating system 100 to represent that an application may leverage the resources of the operating system 100. An application may also leverage the resources of another application. It should be understood that a variety of applications or clients may communicate with the user feedback mechanism 200. As such, the applications and clients disclosed herein are

illustrative and not exhaustive. With the exception of reference numerals in the description of Figure 6, applications and clients are used herein interchangeably. The client 612, such as one represented by any of monitors 500-508 in Figure 6, communicates with the mini LCD 204 through the LCD interface driver 118 or the system BIOS 102. The LCD interface driver 118 is coupled to the LCD interface 600 and to input/output (I/O) logic 608. In a disclosed embodiment, the LCD interface driver 118 is configured as an RS-232 or other serial port driver. In addition, in a disclosed embodiment, the driver 118 includes entry points typical of a Windows driver model (WDM)/NT driver as well as driver entry points specialized for the user feedback mechanism 200. The table below shows a variety of driver entry points used in connection with the user feedback mechanism 200. Beside the name of each driver entry point shown, a description of the driver entry point is provided.

Driver Entry Point Name	**Driver Entry Point Description**
CPQLCD_DriverEntry	This routine ensures that the driver is loaded when the mini LCD is found. It also registers the rest of the entry points of the driver with the operating system and makes them visible to the operating system.
CPQLCD_Dispatch	This routine services the calls from applications/clients and services IOCTLs which are supported by the driver.
CPQLCD_StartIo	This routine is used to interface with a physical device.
CPQLCD_Unload	This routine disconnects and un-registers from all the interrupts that it services once it has unloaded.
CPQLCD_AddDevice	This routine adds a device object when a new device (such as the mini LCD) is detected.
CPQLCD_RemoveDevice	This routine removes a device.
CPQLCD_Create	This routine creates a handle to call into the driver.
CPQLCD_Close	This routine closes a handle for a specific client and deletes the resources allocated by that client.
CPQLCD_Write	This routine handles write requests for the driver.
CPQLCD_Read	This routine handles read requests for the

Driver Entry Point Name	Driver Entry Point Description
	driver.
CPQLCD_ConnectInterrupt	This routine registers the driver with the operating system for servicing specific interrupts from the mini LCD.
CPQLCD_InterruptServiceRoutine	This routine saves data necessary for servicing an interrupt after an interrupt arrives and then dismisses the interrupt.
CPQLCD_DpcRoutine	This routine generates deferred procedure calls queued by the interrupt service routine from any interrupt-related processing.
CPQLCD_Is_LCD_Present	This routine is used to determine if the mini LCD exists.
CPQLCD_Set_Baud_Rate	This routine is used to set the baud rate of the mini LCD.
CPQLCD_SendCmdTo_LCD	This routine is used to send a command to the mini LCD.
GetRegistryDword	This routine is used to read a DWORD.
GetNextIconLocation	This routine routines the next available free icon location for the mini LCD.
CleanUpClientIcons	This routine is used to clean up or clear icons for a particular client or application.
ClearIconLocation	This routine allows a client or application to selectively call and remove icons.
SendAByte	This routine is used to send a byte to a particular port.
SetBaudRate	This routine is used to set the baud rate for a port.
EnableIRQTriggering	This routine is used to enable interrupt request triggering.
ResetIRQTriggering	This routine is used to reset interrupt request triggering.
ReadStatusPort	This routine is used to read a status port.
ReceiveByte	This routine is used to read a byte from a port.
SendByteWhenReady	This routine is used to send a byte when a

Driver Entry Point Name	Driver Entry Point Description
	port is ready.
CompleteIRP	This routine is used to complete an IRP.
StartPacket	This routine is used to queue an IRP. For example, the IRP may be a time command, text command or LED command.
APPCommandComplete	This routine is used to track the last command that was set for the mini LCD.
APPTextCmdFinish	This routine is used to ensure that a command for the mini LCD is completed.
SendTextCommands	This routine is used to send a text command to the mini LCD.
ControlLED	This routine is used to turn on or turn off an LED of the user feedback mechanism.
ResetLCD	This routine is used to reset the mini LCD.

The IOCTLs (name and description) supported by the driver 118 are shown in the table below:

IOCTL Name	IOCTL Description
CPQLCD_GET_CLIENT_ID	This IOCTL is used to fetch a unique identifier for a client/application. The identifier may be used to keep track of any icons downloaded by that client/application.
CPQLCD_CLIENT_CLEAN_UP	This IOCTL is used when a client/application no longer needs the services of the mini LCD and the driver.
CPQLCD_GET_DRIVER_INFO	This IOCTL is used to obtain the name and version number for the driver.
CPQLCD_CLEAR_ICON_BMP	This IOCTL is used for clearing specific bitmaps no longer needed by the client.
CPQLCD_DEF_ICON_BMP	This IOCTL defines and downloads a bitmap and returns an identification number for the icon to uniquely identify that icon.
CPQLCD_DISPLAY_ICON	This IOCTL displays an icon identified by the icon identification number.
CPQLCD_DEFINE_FONT	This IOCTL defines a bitmap for a specific

IOCTL Name	IOCTL Description
	character.
CPQLCD_DISPLAY_TEXT	This IOCTL is used to display text for a specified time on the mini LCD.
CPQLCD_CONTROL_LED	This IOCTL is used to control the state of the LEDs of the user feedback mechanism.
CPQLCD_SET_CLOCK	This IOCTL is used to set the clock of the user feedback mechanism to a desired time.
CPQLCD_QUERY_REVISION	This IOCTL is used to obtain a revision number of the mini LCD for identification purposes.

As shown by some of the IOCTLs above, the driver 118 can be used to dynamically download text or an icon for an application or client. In a disclosed embodiment, the driver 118 is a Plug'n'Play driver. One potential use of the CPQLCD_DEFINE_TEXT IOCTL is to define a new font for text during the clock mode of the user feedback mechanism 200. It should be understood that the driver entry points and IOCTLs disclosed herein are illustrative and not exhaustive. It is noted that the LCD interface 600 enables an application or client component to drive the mini LCD 204 such that an application or client component is not required to communicate directly with the LCD interface driver 118. The LCD interface 600 uses the driver entry points to shield individual applications from having to talk to the LCD interface driver 118 directly.

The input/output logic 608 includes ACPI logic 210 and a universal asynchronous receiver/transmitter (UART) 606. The ACPI logic 210 is coupled to the system BIOS 102 and to the power supply 214. Independent of both the operating system 100 and the LCD interface driver 118, system level interfacing by the client 612 with the mini LCD 204 may be handled through the system BIOS 102. Handshaking and other communication between the LCD interface driver and system BIOS is described in a commonly-assigned U.S. patent application, entitled "OPERATING SYSTEM HANG DETECTION AND CORRECTION," previously incorporated herein. System BIOS 102 and the LCD interface driver 118 are independent of the operating system 100 in a different sense. System BIOS 102 is independent of the operating system 100 in the sense that device states of the primary devices are detected during system initialization (e.g., POST) of the computing system by bypassing the operating system 100. The LCD interface driver 118 and the application 602 or 604 are independent of the operating system 100 in the sense that the manner for the application 602

or 604 and the LCD interface driver 118 to drive the mini LCD 204 is not limited to a particular operating system. The resources or mechanisms of any given operating system may be used in driving the mini LCD 204.

UART-to-UART communication is accomplished through the interface controller 202 between the UART 606 of the I/O logic 608 and a UART 610 of the mini LCD 204. It should be understood that functionality for communicating an operating condition to the mini LCD 204 may be encapsulated in the client 612 or the application 602 or 604. In an alternative embodiment, such functionality may to some extent be integrated into the operating system 100 itself, although potentially sacrificing some independence from the operating system 100. In a disclosed embodiment, the interface controller 202 is a microcontroller in the 80C52 family. For sake of clarity, certain well known components (e.g., memory, latch and other logic) used by a display controller in communicating with a display are not shown.

Thus, a computing system provides a user feedback or digital dashboard mechanism to monitor and alert a user of operating conditions independently of an operating system. An application or client monitors operating conditions and talks to a display panel through a display interface. In this way, the application or client may talk to the display panel in connection with any operating system. During system initialization of the computing system, system BIOS is available for the client to use to talk to the display panel independently of both the display interface and the operating system.

One advantage of the user feedback mechanism is that potentially useful information known by the operating system is actually retrieved, processed and displayed to a user in a comprehensible and user friendly fashion. A further advantage of the user feedback mechanism is that information which is not specifically known by the operating system can be monitored and displayed to a user. The user feedback mechanism provides a comprehensible way to alert a user to useful information which in the past has not been generated or communicated to the user.

By displaying information on the mini display of the user feedback mechanism, useful or valuable feedback as to the condition of the computing system may be provided to a user even while a monitor is off or a screen saver is on. In other words, information which cannot be presented on the main display can be presented on the mini display of the user feedback mechanism. Since the user feedback mechanism operates independently of an operating system, a user can be presented with useful feedback information during system initialization when the operating system is unavailable. Because of its independence from the operating

system, the user feedback mechanism is a guaranteed, failsafe interface to a computing system that users can depend upon for valuable operational and fault tolerant information.

The foregoing disclosure and description of various embodiments are illustrative and explanatory thereof, and various changes in the software components, operating system, applications, clients, notification messages, fault conditions, operating events, signals, icons, text messages, network protocols, order of steps and the like, as well as in the details of the illustrated hardware and software and construction and method of operation may be made without departing from the spirit of the invention.